Getting started with ASP.NET Core

Keywords:

"Choosing to learn and develop with a new framework"

"important to establish early on whether it's right for you"

"background about ASP.NET Core"

"what it is, how it works, and why you should consider it"

"help you to understand the .NET landscape"

"whether now is the right time to consider moving your focus to .NET Core and .NET 5.0"

"advantages ASP.NET Core can offer over previous versions of ASP.NET"

"a good overview of the .NET landscape"

"the role of .NET 5.0"

"the basic mechanics of how ASP.NET Core works"

Questions:

Q: What is important to establish early on when choosing to learn and develop with a new framework?

A: Whether it's right for you.

Q: What background information does the chapter provide about ASP.NET Core?

A: What it is, how it works, and why you should consider it for building your web applications.

Q: Who will benefit from the chapter's help in understanding the .NET landscape?

A: Those who are new to .NET development.

Q: What guidance does the chapter provide for existing .NET developers?

A: Whether now is the right time to consider moving your focus to .NET Core and .NET 5.0 and the advantages ASP.NET Core can offer over previous versions of ASP.NET.

Q: What should readers have by the end of the chapter?

A: A good overview of the .NET landscape, the role of .NET 5.0, and the basic mechanics of how ASP.NET Core works.

An introduction to ASP.NET Core

Keywords:

"ASP.NET Core is a cross-platform, open source, web application framework"

"quickly build dynamic, server-side rendered applications"

"create HTTP APIs that can be consumed by mobile applications"

"browser-based single-page applications such as Angular and React"

"ASP.NET Core provides structure, helper functions, and a framework for building applications"

"ASP.NET Core framework code then calls into your 'handlers'"

"business logic is the core of your application"

"interact with other services such as databases or remote APIs"

"The reasons for using a web framework"

"The previous ASP.NET framework's benefits and limitations"

"What ASP.NET Core is and its motivations"

"why ASP.NET Core was created, its design goals, and why you might want to use it"

Questions:

Q: What type of framework is ASP.NET Core described as?

A: A cross-platform, open source, web application framework.

Q: What kinds of applications can ASP.NET Core quickly build?

A: Dynamic, server-side rendered applications.

Q: Besides web applications, what else can ASP.NET Core be used to create?

A: HTTP APIs that can be consumed by mobile applications, browser-based single-page applications such as Angular and React, or other backend applications.

Q: What does ASP.NET Core provide that saves you from writing a lot of code yourself?

A: Structure, helper functions, and a framework for building applications.

Q: In the ASP.NET Core framework, what does the framework code call into?

A: Your "handlers."

Q: What is described as the core of your application?

A: The business logic.

Q: What can the business logic interact with?

A: Other services such as databases or remote APIs.

Q: What topics are covered in this section about ASP.NET Core?

A: The reasons for using a web framework, the previous ASP.NET framework's benefits and limitations, and what ASP.NET Core is and its motivations.

Q: By the end of the section, what should the reader understand about ASP.NET Core?

A: Why ASP.NET Core was created, its design goals, and why you might want to use it.

Using a web framework

Keywords:

"daunting moving into an area with so many buzzwords and a plethora of ever-changing products"

"perfectly possible to build a static web application without the use of a web framework"

"its capabilities will be limited"

"Just as desktop or mobile development frameworks can help you build native applications"

"ASP.NET Core makes writing web applications faster, easier, and more secure"

"libraries for common things like creating dynamically changing web pages"

"letting users log in to your web app"

"letting users use their Facebook account to log in to your web app using OAuth"

"providing a common structure for building maintainable applications"

"the key to any modern web application is the ability to generate dynamic web pages"

"a dynamic web page may display different data depending on the current logged-in user"

"Without a dynamic framework, it wouldn't be possible to log in to websites or to display any sort of personalized data on a page"

"websites like Amazon, eBay, and Stack Overflow wouldn't be possible"

Questions:

Q: Why can moving into web development be daunting for newcomers?

A: Because there are so many buzzwords and a plethora of ever-changing products.

Q: Is it possible to build a static web application without a web framework?

A: Yes, it is perfectly possible, but its capabilities will be limited.

Q: What happens to the simplicity of building a static web application when adding security or dynamism?

A: The original simplicity will fade before your eyes.

Q: How does ASP.NET Core compare to building everything from scratch?

A: ASP.NET Core makes writing web applications faster, easier, and more secure.

Q: What are some common things ASP.NET Core libraries provide?

A: Creating dynamically changing web pages, letting users log in, using Facebook OAuth login, providing a common structure for maintainable applications, reading configuration files, serving image files, and logging requests.

Q: What is described as the key to any modern web application?

A: The ability to generate dynamic web pages.

Q: What can a dynamic web page do depending on the current logged-in user?

A: Display different data.

Q: What would not be possible without a dynamic framework?

A: Logging in to websites or displaying personalized data on a page.

Q: Which websites are given as examples that wouldn’t be possible without dynamic frameworks?

A: Amazon, eBay, and Stack Overflow.

The benefits and limitations of ASP.NET

Keywords:

"ASP.NET Core is the latest evolution of Microsoft's popular ASP.NET web framework"

"focusing on high developer productivity and prioritizing backwards compatibility"

"significant architectural changes that rethink the way the web framework is designed and built"

"ASP.NET Core is a new framework"

"the whole technology stack has been rewritten"

"ASP.NET should be able to hold its head high when measured against other modern frameworks"

"existing .NET developers should continue to have a sense of familiarity"

"benefits and limitations of the previous ASP.NET web framework"

"ASP.NET Web Forms allowed developers to rapidly create web applications using a graphical designer and a simple event model"

"ASP.NET Web Forms suffered from many issues, especially when building larger applications"

"a lack of testability, a complex stateful model, and limited influence over the generated HTML"

"Microsoft released the first version of ASP.NET MVC in 2009, based on the Model-View-Controller pattern"

"ASP.NET MVC has been through four more iterations since its first release"

"built on the same underlying framework provided by the System.Web.dll file"

"System.Web.dll file is part of .NET Framework, so it comes pre-installed with all versions of Windows"

"ASP.NET framework is a reliable, battle-tested platform"

"reliance is limiting—changes to the underlying System.Web.dll file are far-reaching and, consequently, slow to roll out"

"explicit coupling with the Windows web host, Internet Information Service (IIS)"

".NET Framework to be 'done.' It won't be removed or replaced, but it also won't receive any new features"

"ASP.NET based on System.Web.dll will not receive new features or updates"

"cross-platform web frameworks that can run on Windows as well as Linux and macOS"

"Microsoft felt the time had come to create a framework that was no longer tied to its Windows legacy"

"thus ASP.NET Core was born"

Questions:

Q: When was ASP.NET Core released?

A: ASP.NET Core was released in June 2016.

Q: What did previous versions of ASP.NET focus on?

A: High developer productivity and prioritizing backwards compatibility.

Q: How does ASP.NET Core differ from previous versions of ASP.NET?

A: It makes significant architectural changes that rethink the way the web framework is designed and built.

Q: What has been rewritten in ASP.NET Core?

A: The whole technology stack, including both the web framework and the underlying platform.

Q: What is the philosophy behind ASP.NET Core's design?

A: ASP.NET should hold its head high against other modern frameworks while giving existing .NET developers a sense of familiarity.

Q: What were some issues with ASP.NET Web Forms?

A: Lack of testability, a complex stateful model, and limited influence over the generated HTML.

Q: What pattern is ASP.NET MVC based on?

A: The Model-View-Controller pattern.

Q: On what underlying framework is ASP.NET MVC built?

A: The System.Web.dll file.

Q: What is a limitation of the ASP.NET framework's reliance on System.Web.dll?

A: Changes to System.Web.dll are far-reaching and slow to roll out, limiting ASP.NET's ability to evolve.

Q: What explicit coupling limits ASP.NET to Windows platforms?

A: Coupling with the Windows web host, Internet Information Service (IIS).

Q: What is Microsoft's current stance on the .NET Framework?

A: It is "done" and will not receive new features or updates.

Q: Why was ASP.NET Core created?

A: To create a framework no longer tied to its Windows legacy and to support cross-platform development.

What is ASP.NET Core?

Keywords:

"four main goals"

"To be run and developed cross-platform"

"To have a modular architecture for easier maintenance"

"To be developed completely as open source software"

"To be applicable to current trends in web development"

"ASP.NET development had always been focused, and dependent, on the Windows-only .NET Framework"

"Microsoft created a lightweight platform that runs on Windows, Linux, and macOS called .NET Core (and subsequently .NET 5.0)"

".NET 5.0 is the next version of .NET Core after 3.1"

".NET 5.0 represents a unification of .NET Core and other .NET platforms into a single runtime and framework"

".NET Core (and its successor, .NET 5.0) employs many of the same APIs as .NET Framework"

".NET Core is more modular and only implements a subset of the features .NET Framework does"

".NET Core is a completely separate platform, rather than a fork of .NET Framework"

"With .NET 5.0 alone, it's possible to build console applications that run cross-platform"

"ASP.NET Core to be an additional layer on top of console applications"

"ASP.NET Core is composed of many small libraries that you can choose from"

"Some of the libraries are common and will appear in virtually every application"

"Other libraries build on top of these base capabilities to provide application-specific functionality"

"Most of the libraries you'll use in ASP.NET Core can be found on GitHub"

"All ASP.NET Core applications will follow a similar design for basic configuration"

"the framework is flexible, leaving you free to create your own code conventions"

"These common libraries, the extension libraries that build on them, and the design conventions they promote are covered by the somewhat nebulous term ASP.NET Core"

Questions:

Q: What are the four main goals that motivated the development of ASP.NET Core?

A: To be run and developed cross-platform, to have a modular architecture for easier maintenance, to be developed completely as open source software, and to be applicable to current trends in web development.

Q: What platform did Microsoft create to support ASP.NET Core that runs on multiple operating systems?

A: Microsoft created a lightweight platform called .NET Core (and subsequently .NET 5.0) that runs on Windows, Linux, and macOS.

Q: What does .NET 5.0 represent in relation to .NET Core and other .NET platforms?

A: .NET 5.0 represents a unification of .NET Core and other .NET platforms into a single runtime and framework.

Q: How does .NET Core differ from .NET Framework in terms of features?

A: .NET Core is more modular and only implements a subset of the features .NET Framework does.

Q: What kind of applications can be built with .NET 5.0 alone?

A: Console applications that run cross-platform.

Q: How does ASP.NET Core relate to console applications in .NET 5.0?

A: ASP.NET Core is an additional layer on top of console applications to convert them into web applications by adding and composing libraries.

Q: What is the nature of the libraries composing ASP.NET Core?

A: ASP.NET Core is composed of many small libraries that you can choose from, adding only what you need.

Q: Where can most of the libraries used in ASP.NET Core be found?

A: On GitHub, in the Microsoft ASP.NET Core organization repositories at https://github.com/dotnet/aspnetcore.

Q: What flexibility does ASP.NET Core offer regarding application configuration and code conventions?

A: The framework is flexible, allowing you to create your own code conventions, though applications will follow a similar design for basic configuration as suggested by common libraries.

When to choose ASP.NET Core

Keywords:

"should you use it?"

"Microsoft is recommending that all new .NET web development should use ASP.NET Core"

"switching to or learning a new web stack is a big ask for any developer or company"

"What sort of applications you can build with ASP.NET Core"

"Some of the highlights of ASP.NET Core"

"Why you should consider using ASP.NET Core for new applications"

"Things to consider before converting existing ASP.NET applications to ASP.NET Core"

Questions:

Q: What is Microsoft's recommendation for new .NET web development?

A: Microsoft is recommending that all new .NET web development should use ASP.NET Core.

Q: What is a challenge mentioned about switching to or learning ASP.NET Core?

A: Switching to or learning a new web stack is a big ask for any developer or company.

Q: What topics does the section cover regarding ASP.NET Core?

A: The section covers what sort of applications you can build with ASP.NET Core, some highlights of ASP.NET Core, why you should consider using it for new applications, and things to consider before converting existing ASP.NET applications.

What type of applications can you build?

Keywords:

"ASP.NET Core provides a generalized web framework"

"building rich, dynamic websites"

"open source content management system (CMS) Orchard"

"cloudscribe CMS project was written specifically for ASP.NET Core"

"Traditional page-based server-side-rendered web applications"

"single-page applications (SPAs)"

"client-side framework that commonly talks to a REST server"

"REST stands for representational state transfer"

"create a web service or remote procedure call (RPC)-style service"

"your application might expose only a single endpoint"

"ASP.NET Core is perfectly designed for building simple services"

"Microsoft has pledged to provide full support for Long Term Support (LTS) versions"

"you have two primary dimensions to consider: whether you're already a .NET developer, and whether you're creating a new application or looking to convert an existing one"

Questions:

Q: What types of applications can ASP.NET Core be used to build?

A: ASP.NET Core can be used for building rich, dynamic websites, including e-commerce sites, content-based sites, or large n-tier applications.

Q: What is an example of a content management system that was redeveloped to run on ASP.NET Core?

A: The open source content management system Orchard has been redeveloped as Orchard Core to run on ASP.NET Core.

Q: What does REST stand for and what does it typically use?

A: REST stands for representational state transfer and typically uses lightweight and stateless HTTP calls.

Q: Can ASP.NET Core create services other than RESTful ones?

A: Yes, it's easy to create a web service or remote procedure call (RPC)-style service with ASP.NET Core.

Q: What is one feature of ASP.NET Core that makes it suitable for building simple services?

A: ASP.NET Core is perfectly designed for building simple services thanks to its cross-platform support and lightweight design.

Q: How long has Microsoft pledged to support Long Term Support (LTS) versions of .NET Core and ASP.NET Core?

A: Microsoft has pledged to provide full support for Long Term Support (LTS) versions for at least three years from the time of their release.

Q: What are the two primary dimensions to consider when deciding whether to use ASP.NET Core?

A: Whether you're already a .NET developer, and whether you're creating a new application or looking to convert an existing one.

If you're new to .NET development

Keywords:

"ASP.NET Core as an attractive option for web development beginners"

"modern, high-performance, open source web framework"

"familiar design patterns and paradigms"

"build and run on any platform"

"ASP.NET Core is a re-imagining of the ASP.NET framework"

".NET Core has several years of widespread production use"

"mature, stable, and reliable .NET Framework"

"Model-View-Controller (MVC) pattern"

"Node.js is known for the way it processes requests using small discrete modules (called a pipeline)"

"dependency injection is found in a wide variety of frameworks"

"primary language of .NET development, and ASP.NET Core in particular, is C#"

"C# is an object-oriented C-based language"

"C# language is also designed in the open on GitHub"

"ability to develop and run on any platform"

"wide range of distributions are supported (RHEL, Ubuntu, Debian, CentOS, Fedora, and openSUSE)"

"ASP.NET Core even runs on the tiny Alpine distribution"

Questions:

Q: What is ASP.NET Core promoted as for web development beginners?

A: ASP.NET Core is promoted as an attractive option for web development beginners.

Q: What type of web framework is ASP.NET Core described as?

A: ASP.NET Core is described as a modern, high-performance, open source web framework.

Q: What languages can you use with ASP.NET Core?

A: You can use C#, VB.NET, or F# with ASP.NET Core.

Q: What platform does ASP.NET Core run on?

A: You can build and run ASP.NET Core on any platform.

Q: What is ASP.NET Core a re-imagining of?

A: ASP.NET Core is a re-imagining of the ASP.NET framework.

Q: How long has .NET Core had widespread production use?

A: .NET Core has several years of widespread production use.

Q: What mature framework has influenced .NET Core?

A: The mature, stable, and reliable .NET Framework has influenced .NET Core.

Q: Which design patterns commonly used in other frameworks are also part of ASP.NET Core?

A: Model-View-Controller (MVC), pipelines for processing requests, and dependency injection.

Q: What is the primary language used in ASP.NET Core development?

A: The primary language used is C#.

Q: What are some powerful features of C# mentioned in the passage?

A: Language Integrated Query (LINQ), closures, and asynchronous programming constructs.

Q: Where is the C# language designed?

A: The C# language is designed in the open on GitHub.

Q: Which operating systems support ASP.NET Core development and deployment?

A: Mac, Windows, and Linux support ASP.NET Core development and deployment.

Q: Name some Linux distributions supported by ASP.NET Core.

A: RHEL, Ubuntu, Debian, CentOS, Fedora, and openSUSE.

Q: What lightweight Linux distribution does ASP.NET Core run on for compact container deployments?

A: ASP.NET Core runs on the tiny Alpine distribution.

Built with containers in mind

Keywords:

"Traditionally, web applications were deployed directly to a server"

"virtual machines allow operating systems to be installed in a layer of virtual hardware"

"easy maintenance, deployment, and recovery"

"they're also heavy, both in terms of file size and resource use"

"containers are far more lightweight and don't have the overhead of virtual machines"

"built in a series of layers and don't require you to boot a new operating system"

"quick to start and are great for quick provisioning"

"containers, and Docker in particular, are quickly becoming the go-to platform for building large, scalable systems"

"containers have never been a particularly attractive option for ASP.NET applications"

"with ASP.NET Core, .NET 5.0, and Docker for Windows, that's all changing"

"a lightweight ASP.NET Core application running on the cross-platform .NET 5.0 framework is perfect for thin container deployments"

"learn more about your deployment options in chapter 16"

Questions:

Q: How were web applications traditionally deployed?

A: Traditionally, web applications were deployed directly to a server.

Q: What is one advantage of virtual machines over direct installation?

A: Virtual machines allow easy maintenance, deployment, and recovery.

Q: What is a disadvantage of virtual machines mentioned in the passage?

A: They're heavy in terms of file size and resource use.

Q: How do containers differ from virtual machines in terms of overhead?

A: Containers are far more lightweight and don't have the overhead of virtual machines.

Q: Why are containers quick to start?

A: Because they're built in layers and don't require booting a new operating system.

Q: What platform is becoming the go-to for building large, scalable systems?

A: Containers, and Docker in particular, are becoming the go-to platform.

Q: How has the attractiveness of containers for ASP.NET applications changed?

A: Containers have never been attractive for ASP.NET, but with ASP.NET Core, .NET 5.0, and Docker for Windows, that's all changing.

Q: What kind of ASP.NET Core application is ideal for thin container deployments?

A: A lightweight ASP.NET Core application running on the cross-platform .NET 5.0 framework.

Q: Where can you learn more about deployment options?

A: You can learn more about deployment options in chapter 16.

If you’re a .NET Framework developer creating a new application

Keywords:

".NET developer"

"Early versions of .NET Core were lacking in some features"

".NET Core 3.1 and .NET 5.0"

"Microsoft now explicitly advises that all new .NET applications should use .NET 5.0"

"Microsoft has pledged to provide bug and security fixes for the older ASP.NET framework"

"won’t provide any more feature updates"

"Cross-platform development and deployment"

"A focus on performance as a feature"

"A simplified hosting model"

"Regular releases with a shorter release cycle"

"Open source"

"Modular features"

"ability to build and deploy cross-platform opens the door to a whole new avenue of applications"

"cheaper Linux VM hosting in the cloud"

"use Docker containers for repeatable continuous integration"

"write .NET code on your Mac without needing to run a Windows virtual machine"

".NET Core and .NET 5.0 are inherently cross-platform"

"Windows-specific features like the Registry or Directory Services can be enabled with a compatibility pack"

"hosting model for the previous ASP.NET framework was a relatively complex one, relying on Windows IIS"

"alternative hosting model has been adopted"

"Kestrel: a fast, cross-platform HTTP server on which ASP.NET Core can run"

"ASP.NET Core applications are console applications that self-host a web server and handle requests directly"

"optionally run ASP.NET Core inside of IIS"

"Changing the hosting model to use a built-in HTTP web server has created another opportunity"

"Performance has been somewhat of a sore point for ASP.NET applications in the past"

"TechEmpower announced that ASP.NET Core with Kestrel was the fastest of over 400 frameworks tested"

"performance improvements made to Kestrel did not come from the ASP.NET team members themselves"

"Developing in the open means you typically see fixes and features make their way to production faster"

".NET 5.0, and hence ASP.NET Core, is designed to be released in small increments"

"new version every year, and a new Long Term Support (LTS) version released every two years"

"Additional functionality is provided as NuGet packages"

"ASP.NET Core is highly modular"

"pay-for-play approach to dependencies"

"Middleware “pipeline” for defining your application’s behavior"

"Built-in support for dependency injection"

"Combined UI (MVC) and API (Web API) infrastructure"

"Highly extensible configuration system"

"Scalable for cloud platforms by default using asynchronous programming"

"Microsoft fully supports ASP.NET Core"

"largest obstacle you’re likely to come across is wanting to use programming models that are no longer supported in ASP.NET Core"

Questions:

Q: What has Microsoft explicitly advised regarding new .NET applications?

A: Microsoft now explicitly advises that all new .NET applications should use .NET 5.0.

Q: Will Microsoft provide feature updates for the older ASP.NET framework?

A: No, Microsoft won’t provide any more feature updates for the older ASP.NET framework.

Q: What are some main benefits of ASP.NET Core over the previous ASP.NET framework?

A: Cross-platform development and deployment, a focus on performance as a feature, a simplified hosting model, regular releases with a shorter release cycle, open source, and modular features.

Q: What new avenues does cross-platform development open for .NET developers?

A: It opens the door to cheaper Linux VM hosting in the cloud, use of Docker containers for repeatable continuous integration, and writing .NET code on a Mac without needing a Windows virtual machine.

Q: How can Windows-specific features be used with .NET 5.0?

A: Windows-specific features like the Registry or Directory Services can be enabled with a compatibility pack in .NET 5.0.

Q: What was the hosting model for the previous ASP.NET framework?

A: It was a relatively complex model relying on Windows IIS to provide web server hosting.

Q: What hosting model has ASP.NET Core adopted?

A: ASP.NET Core uses an alternative hosting model with Kestrel, a fast, cross-platform HTTP server.

Q: How do ASP.NET Core applications handle requests in the new hosting model?

A: ASP.NET Core applications are console applications that self-host a web server and handle requests directly.

Q: Can ASP.NET Core run inside IIS?

A: Yes, ASP.NET Core can optionally run inside IIS, which can have performance benefits.

Q: How has performance been addressed in ASP.NET Core?

A: The ASP.NET team focused on making the Kestrel HTTP server as fast as possible, and TechEmpower benchmarks showed it was the fastest of over 400 frameworks tested.

Q: Who contributed many performance improvements to Kestrel?

A: Many performance improvements came from contributors to the open source project on GitHub, not just the ASP.NET team.

Q: How often are new versions of .NET 5.0 and ASP.NET Core released?

A: A new version is released every year, with a new Long Term Support (LTS) version every two years.

Q: How is additional functionality provided in ASP.NET Core?

A: Additional functionality is provided as NuGet packages, independent of the .NET 5.0 platform.

Q: What is the approach to dependencies in ASP.NET Core?

A: ASP.NET Core uses a pay-for-play approach, starting with a bare-bones application and adding only required libraries.

Q: Name some key infrastructure improvements in ASP.NET Core.

A: Middleware pipeline, built-in support for dependency injection, combined UI (MVC) and API infrastructure, highly extensible configuration system, and scalability for cloud platforms using asynchronous programming.

Q: What is a significant obstacle for developers when switching to ASP.NET Core?

A: Wanting to use programming models that are no longer supported in ASP.NET Core, such as Web Forms or WCF server.

Converting an existing ASP.NET application to ASP.NET Core

Keywords:

"existing application is presumably already providing value"

"advantages of adopting ASP.NET Core are much the same as for new applications"

"cross-platform deployment, modular features, and a focus on performance"

"Your application uses ASP.NET Web Forms"

"Your application is built using WCF"

"Your application is large, with many “advanced” MVC features"

"Web Forms is inextricably tied to System.Web.dll"

"Converting an application to ASP.NET Core would effectively involve rewriting the application from scratch"

"slowly introduce Web API concepts and try to reduce the reliance on legacy Web Forms constructs such as ViewData"

"Windows Communication Foundation (WCF) is only partially supported in ASP.NET Core"

"possible to consume some WCF services, but support is spotty at best"

"no supported way to host a WCF service from an ASP.NET Core application"

"consider using gRPC instead"

"gRPC is a modern RPC framework with many concepts similar to WCF"

"ASP.NET Core is built with many similar features to the previous version of ASP.NET MVC, but the underlying architecture is different"

"porting an application from ASP.NET MVC to ASP.NET Core is at least as big a rewrite as porting from ASP.NET Web Forms to ASP.NET MVC"

"Microsoft will support .NET Framework for the foreseeable future"

"best opportunity for getting started is on small, green-field, new projects instead of existing applications"

"if the existing application in question is small or will need significant future development, then porting may be a good option"

"work in small iterations where possible"

"application consists primarily of MVC or Web API controllers and associated Razor views"

Questions:

Q: What are the advantages of adopting ASP.NET Core for existing applications?

A: Cross-platform deployment, modular features, and a focus on performance.

Q: Which types of applications are clear indicators against converting to ASP.NET Core?

A: Applications that use ASP.NET Web Forms, are built using WCF, or are large with many advanced MVC features.

Q: Why is converting a Web Forms application to ASP.NET Core not advisable?

A: Because Web Forms is inextricably tied to System.Web.dll and would require rewriting the application from scratch.

Q: What approach is recommended instead of converting a Web Forms application directly?

A: Slowly introduce Web API concepts and reduce reliance on legacy Web Forms constructs such as ViewData.

Q: How is WCF supported in ASP.NET Core?

A: WCF is only partially supported; some services can be consumed but support is spotty, and hosting WCF services is not supported.

Q: What alternative does the passage suggest if WCF's RPC-style programming is needed?

A: Consider using gRPC, which is supported by ASP.NET Core out of the box.

Q: Why might porting a complex MVC application to ASP.NET Core be difficult?

A: Because ASP.NET Core has a different underlying architecture and some previous features don't have direct replacements.

Q: How does Microsoft compare porting from ASP.NET MVC to ASP.NET Core versus from Web Forms to MVC?

A: Microsoft suggests porting from ASP.NET MVC to ASP.NET Core is at least as big a rewrite as porting from Web Forms to MVC.

Q: When does the passage suggest not converting an application to ASP.NET Core?

A: If the application is rarely used, isn't part of core business, or won't need significant development soon.

Q: What is the best opportunity for starting with ASP.NET Core according to the passage?

A: Small, green-field, new projects instead of existing applications.

Q: Under what conditions might porting an existing application to ASP.NET Core be a good option?

A: If the existing application is small or will need significant future development.

Q: What is the recommended method for porting an application?

A: To work in small iterations rather than converting the entire application at once.

Q: What type of applications may benefit most from moving to ASP.NET Core?

A: Applications consisting primarily of MVC or Web API controllers and associated Razor views.

How does ASP.NET Core work?

Keywords:

"how does ASP.NET Core work"

"what ASP.NET Core is and the sort of applications you should use it for"

"how an application built with ASP.NET Core works"

"user requesting a URL to a page being displayed in the browser"

"how an HTTP request works for any web server"

"how ASP.NET Core extends the process to create dynamic web pages"

Questions:

Q: What does the passage aim to explain about ASP.NET Core?

A: How an application built with ASP.NET Core works.

Q: What is the starting point of the ASP.NET Core application process described in the passage?

A: The user requesting a URL.

Q: What is the endpoint of the ASP.NET Core application process described in the passage?

A: A page being displayed in the browser.

Q: What two things will the reader learn in this section according to the passage?

A: How an HTTP request works for any web server, and how ASP.NET Core extends the process to create dynamic web pages.

How does an HTTP web request work?

Keywords:

"ASP.NET Core is a framework for building web applications that serve data from a server"

"building a web app that you can view in a web browser"

"user navigates to a website or types a URL in their browser"

"hostname and a path to some resource on the web app"

"using the HTTP protocol"

"hostname of a website uniquely identifies its location on the internet"

"request is potentially received and rebroadcast at multiple routers"

"request is processed"

"server receives the request"

"generate an HTTP response"

"server responds with some HTML"

"HTML is added to the HTTP response"

"browser begins receiving the HTTP response"

"HTML page may also reference other pages and links on the server"

"fetching every referenced file"

"all fetched using the exact same HTTP request process"

"interactions that take place on the internet are a facade over this same basic process"

"Amazon.com homepage (www.amazon.com) makes 606 requests"

"see how ASP.NET Core dynamically generates the response on the server"

Questions:

Q: What is ASP.NET Core described as in the passage?

A: A framework for building web applications that serve data from a server.

Q: What common scenario is mentioned for web developers in the passage?

A: Building a web app that you can view in a web browser.

Q: What initiates the request process to a server?

A: A user navigates to a website or types a URL in their browser.

Q: What does the URL consist of according to the passage?

A: A hostname and a path to some resource on the web app.

Q: What protocol is used when sending the request to the server?

A: The HTTP protocol.

Q: What does the hostname of a website uniquely identify?

A: Its location on the internet.

Q: What happens to the request as it travels through the internet?

A: It is potentially received and rebroadcast at multiple routers.

Q: When is the request processed by the server?

A: When it reaches the server associated with the hostname.

Q: What does the server do after receiving a valid request?

A: It will generate an HTTP response.

Q: What kind of response might the server generate?

A: A web page, an image, a JavaScript file, or a simple acknowledgment.

Q: What does the server respond with in the example given?

A: Some HTML.

Q: What does the browser start doing as soon as it begins receiving the HTTP response?

A: Displaying content on the screen.

Q: What must the browser do to display a complete web page?

A: Fetch every referenced file.

Q: What files are fetched using the same HTTP request process?

A: HTML, images, CSS for styling, and JavaScript files.

Q: What is said about interactions on the internet in the passage?

A: They are a facade over this same basic process.

Q: How many requests does the Amazon.com homepage make?

A: 606 requests.

Q: What will the next section explain after describing the request process?

A: How ASP.NET Core dynamically generates the response on the server.

How does ASP.NET Core process a request?

Keywords:

"the same HTTP protocol as before"

"ASP.NET Core itself encompasses everything that takes place on the server"

"verifying that the request is valid, handling login details, and generating HTML"

"built-in web server, Kestrel by default"

"constructing an internal representation of the data, an HttpContext object"

"details stored in HttpContext"

"return an 'access denied' message, or to send an email"

"convert the representation into a raw HTTP response"

"forward it to the user's browser"

"this process appears to be the same as for the generic HTTP request"

"All the differences are server-side, within your application"

"the components that make up a typical ASP.NET Core application"

"generating a response in ASP.NET Core"

"step through an application slowly"

"covering each of the components in detail"

Questions:

Q: What protocol do browsers use to communicate with an ASP.NET Core application?

A: The same HTTP protocol as before.

Q: What does ASP.NET Core encompass on the server?

A: Everything that takes place on the server to handle a request.

Q: What are some of the tasks handled by ASP.NET Core when a request is received?

A: Verifying that the request is valid, handling login details, and generating HTML.

Q: What is the name of the built-in web server in every ASP.NET Core application?

A: Kestrel by default.

Q: What does Kestrel construct from raw requests?

A: An internal representation of the data, an HttpContext object.

Q: What does the application use to generate an appropriate response?

A: The details stored in HttpContext.

Q: What are possible responses generated by the application?

A: Some HTML, an "access denied" message, or to send an email.

Q: What happens after the application processes the request?

A: It will return the response to the web server.

Q: What does the ASP.NET Core web server do with the response?

A: Convert the representation into a raw HTTP response and send it to the network.

Q: Where does the network forward the HTTP response?

A: To the user’s browser.

Q: How does the user perceive the request and response process?

A: As the same as for the generic HTTP request.

Q: Where do all the differences in the request and response process exist?

A: Server-side, within your application.

Q: What topic is covered in part 1 of the book?

A: The components that make up a typical ASP.NET Core application.

Q: What is typically involved in generating a response in ASP.NET Core?

A: A lot goes into generating a response in ASP.NET Core.

Q: How will the book explain the response generation process?

A: Step through an application slowly, covering each of the components in detail.

ASP.NET Core and reverse proxies

Keywords:

"expose ASP.NET Core applications directly to the internet"

"Kestrel receives requests directly from the network"

"use a reverse proxy between the raw network and your application"

"reverse-proxy server will typically be IIS"

"NGINX, HAProxy, or Apache"

"receiving requests and forwarding them to the appropriate web server"

"exposed directly to the internet"

"security and performance for the web servers"

"decoupling of your application from the underlying operating system"

"cross-platform and used behind a variety of proxies"

"change anything else about your application"

"hardened against potential threats from the public internet"

"restarting a process that has crashed"

"Kestrel can remain a simple HTTP server"

"Kestrel is concerned with generating HTTP responses"

"the reverse proxy is concerned with handling the connection to the internet"

Questions:

Q: Can ASP.NET Core applications be exposed directly to the internet?

A: Yes, so that Kestrel receives requests directly from the network.

Q: What is more common than exposing ASP.NET Core directly to the internet?

A: To use a reverse proxy between the raw network and your application.

Q: What is the typical reverse-proxy server on Windows?

A: IIS.

Q: What reverse proxies might be used on Linux or macOS?

A: NGINX, HAProxy, or Apache.

Q: What is a reverse proxy responsible for?

A: Receiving requests and forwarding them to the appropriate web server.

Q: Where is the reverse proxy exposed?

A: Directly to the internet.

Q: What are the benefits of using a reverse proxy?

A: Security and performance for the web servers.

Q: How does a reverse proxy help with platform flexibility?

A: It allows Kestrel to be cross-platform and used behind a variety of proxies.

Q: What can be used in place of Kestrel without changing the application?

A: A new ASP.NET Core web server.

Q: What threat-related advantage does a reverse proxy offer?

A: It can be hardened against potential threats from the public internet.

Q: What extra responsibility can a reverse proxy handle?

A: Restarting a process that has crashed.

Q: Why can Kestrel remain a simple HTTP server behind a reverse proxy?

A: Because it doesn’t have to worry about extra features.

Q: What is Kestrel concerned with?

A: Generating HTTP responses.

Q: What is the reverse proxy concerned with?

A: Handling the connection to the internet.

What you will learn in this book

Keywords:

"an in-depth tour of the ASP.NET Core framework"

"familiar with C# or a similar object-oriented language"

"Basic familiarity with web concepts like HTML and JavaScript"

"create page-based applications with Razor Pages"

"model-binding, validation, and routing"

"generate HTML for web pages using Razor syntax and Tag Helpers"

"dependency injection, configuration, and logging"

"protect your application using security best practices"

"examples are generally small and self-contained"

"focus on a single feature at a time"

"Visual Studio for most of the examples"

"your favorite editor or IDE"

"setting up your editor or IDE and installing the .NET 5.0 SDK"

"everything you see can be achieved equally well on Linux or Mac platforms"

"install .NET 5.0 from https://dotnet.microsoft.com/download"

"configure your development environment for working with ASP.NET Core and .NET 5.0"

"create your first application from a template and run it"

"see how they all work together to render a web page"

Questions:

Q: What does the book provide regarding ASP.NET Core?

A: An in-depth tour of the ASP.NET Core framework.

Q: What prior knowledge is beneficial before reading the book?

A: Familiarity with C# or a similar object-oriented language.

Q: What basic web concepts should readers be familiar with?

A: HTML and JavaScript.

Q: What kind of applications will readers learn to create?

A: Page-based applications with Razor Pages.

Q: What are some key ASP.NET Core concepts covered in the book?

A: Model-binding, validation, and routing.

Q: What syntax and tools are used to generate HTML in the book?

A: Razor syntax and Tag Helpers.

Q: What features will be covered as applications grow more complex?

A: Dependency injection, configuration, and logging.

Q: What best practices will the book teach to protect applications?

A: Security best practices.

Q: What type of examples are used throughout the book?

A: Small and self-contained.

Q: What is the purpose of the small examples in the book?

A: To focus on a single feature at a time.

Q: What development environment does the author use for examples?

A: Visual Studio.

Q: Can readers use tools other than Visual Studio?

A: Yes, your favorite editor or IDE.

Q: What does Appendix A include?

A: Details on setting up your editor or IDE and installing the .NET 5.0 SDK.

Q: On which platforms can the examples be achieved?

A: Equally well on Linux or Mac platforms.

Q: Where can .NET 5.0 be installed from?

A: https://dotnet.microsoft.com/download.

Q: What will the next chapter guide you to do?

A: Create your first application from a template and run it.

Q: What will you see about the application’s components in the next chapter?

A: How they all work together to render a web page.